CSCI 720, Section 03

Report for Project 2: Newsgroup Text Analysis

Synopsis:

Test mining has various applications, one of them being classification of text to categories they belong to. Given a document corpus consisting of news postings, the aim of our project is to train and build a text classifier model which classifies news postings into topics they belong to. Along with classification, our aim also consists of building a model with good level of accuracy and efficiency.

As classification is a form of Supervised learning, we build our model from the training data set. We have divided the input document corpus into three sets of data - training, testing and validation data set. Training and validation data set will be used for evaluation of our model. For this project, we iteratively built 2 models based on K-Nearest Neighbor and Naive Bayes algorithm respectively. We decided Naive Bayes model as our final model. The advantages, limitations, tradeoffs, time and space complexity of our model along with its evaluation and results are also discussed.

Analysis of input data:

The entire document corpus has 20 sub folders, one for each news topic. There are around 1000 news postings under each news topic, with the first part of news article being structured and the rest is free flowing data or unstructured. Overall, there are about 20000 text documents, with about 75 total lines on an average. Thus, memory requirement is high and processing on such a huge amount of data may take a lot of processor time.

Thus, we need to build a model which is memory efficient and also time efficient. In the structured data part of every text document, we need to ignore the newsgroup topic, as this is exactly what we are trying to determine. We noticed that most of the structured data attributes do not add much to the information we need to find.

Cleaning and Preparation Operations performed:

As our aim is to build a model to determine the topic a news post is written about, from the free flowing text of the news post, our only source to gain such information are the words in each text document. The document corpus is huge, and checking for each word is an extremely exhaustive process. Moreover, we need to be sure that checking each word is necessary, and whether it really gives us the output we desire.

After our initial discussions, we realized that there are many words in the text document which can be ignored without affecting the accuracy of our model. These are the stop words like "and", "while", "before" etc. Also, punctuation marks can be ignored. [1][5]

Although the structured part of text can be ignored without affecting the accuracy of our model, the time required to strip the structured data from each of the 20000 documents would add to our time complexity. Also, the structured data is not uniform in every text document, there are different number of attributes for each text document.

We also considered the fact that the only source of knowledge to classify a news posting are words themselves. Thus, we wanted to make sure that we do not lose any important knowledge in the cleaning phase which may be possible as the structured data is inconsistent throughout the corpus.

Thus, we decided we would only perform preliminary cleaning of the data and not spend time in exhaustive cleaning. However, we **have removed** the newsgroup topic by adding the topics in the stop words list, as this is exactly what our project is meant to determine.

Training, Testing and Validation data sets:

We chose 60/20/20 split using the holdout method to split the given input data set.[10]

We observed that all the text documents in each sub-folder of the 20_newsgroups folder do not have any specific pattern to them. If we go in ascending order of the numerically-named documents in each sub-folder, most of the documents are of the same size. Attributes in the structured part of the document are not uniform, with different attributes in different documents. Moreover, as this is free-flowing text, there is no way we can determine differences among documents of the same folder without parsing every word in every document. This would mean complexity of our solution increasing even before we start text mining.

Thus, based on this observation of the data set provided, we decided to use the Holdout Method to build our training, testing and validation data sets. Thus, we have used the first 60% text documents of every sub-folder as our training data set, the next 20% as testing data set and the next 20% as our validation test data set.

Concept of our Solution:

We first thought of ways to solve this problem if we had to do it manually, so that we could translate the same idea into an actual data model. Our approach consists of going through each document and finding 'key' words which seem to be most relevant to the document. A simple way of finding 'relevant word' can be finding the most frequently occurring words in the news post. For example, if the news posting is about sports, in particular, say baseball, words like player, play, game, goal etc occur more frequently as compared to other documents.

However, as the news posting is free-flowing text in the English language, there can be many common words which have no relevance to the topic the article belongs to. For example, most of the article words like 'the', 'and' etc. Thus, we should not consider the words like articles, prepositions etc.[5]

For any new incoming news post, we repeat the same steps we did for training process : find a particular set of most frequent and relevant words for each topic. The news topic with which this value matches determines the class the new sample belongs to.

We need to translate this model into a working model using computing resources and algorithms. From our manual model, we concluded that our manual approach can be implemented using the K-Nearest Neighbor and the Naive Bayes Algorithm. We have used Python tool to build our model. We deliberated using R, Weka or Python and searched for relevant libraries and functions available with each tool. We decided to use the scikit package available in Python which has inbuilt K-Nearest Neighbors and Naive Bayes algorithms as all three members were more comfortable with coding in Python.[2]

Model Built: Model 1, K-Nearest Neighbors

Concept: Our main idea is to build a data structure (list) for each news topic which stores 'm' most frequent words of each news group topic corpus. This would be the "training" phase of the model. Once we have such a list for each news group topic, we use it as a sort of "look-up" table when we try to classify the new incoming news post text.

For the input news posting, after cleaning and processing the data to remove the stop words, we evaluate the entire news. We again find the 'm' most commonly occurring words in a given new data instance. Now we compare how many documents in the training set have the same set of matching. Of all these neighbors, we do a kind of 'voting algorithm'. The majority of class of all these matched documents determines the class of the new data instance.

Determining K:

In a K-Nearest Neighbor Algorithm, 'K' neighbors vote for the new data sample. The class receiving the majority of the votes is the true value of the new data sample. The 'closeness' is determined by a measure of distance.[11] As our data set is free-flowing text, we need to determine this measure of closeness. We decided to use the number of documents having most matching's with new data instance to be this measure of distance. The majority class of these closely matched documents will determine the class value of new data instance.

For example, for every new data instance, we calculate 100 most frequent yet relevant words from the text of the data instance. We compare these top 100 words to each set of top 100 words we determined for each news posting document in our training phase. Let us say we found 10 such 'closely matched' documents. Now, if 6 of these documents belong to a particular class, then by vote if majority, the new data instance will be classified into that class.

Steps to build K-NN model:

1. Read all the documents from the path given, storing into different data structures for training, testing and validation data set.[9]

2. For every input document, do the following:

    1. Data Cleaning and Preparation: Remove the attribute 'newsgroup: ' and 'Xref:' which provide the name of the topic. Remove stopwords, punctuation marks and numbers.
    2. Bag of words: Find word frequency counts for every word in the document. Sort the frequency counts and store the top 'm' values of the count.

3. For every new data instance, again find the top 'm' words occurring in the document. Compare these words with top 'm' words found for every document.

4. Find 'K' nearest neighbors : Find 'K' files whose words most closely match the new data instance.

5. The majority class of these 'K' instances is the class of our new data instance.

Iterations of our model:

For 'm': We iteratively check ran the model with changing values of m and found as value of 'm' increases, accuracy increases. However, for this model, at beyond 100, accuracy was not affected much. Hence we chose '100'.

For 'K': As we increased value of 'K', the accuracy of our model increases. However, time taken to build the model also increases. Hence, we observed a tradeoff.

For the given data set, optimal value of 'm' and 'K' were found to be 100 and 10 respectively.

Model Built: Model 2, Naive Bayes

Concept:

For our project, once we have the 'Bag of Words'[5] for our training data set, we need to determine which news group topic the new incoming data instance belongs to. Naive Bayes uses the concept of Prior and Conditional probabilities to find set of words of new data instance which are most 'probable' instance with those in the training set.[4] The news group topic with the highest probable ownership of the new data instance is determined to be the class value for the new data instance.

Challenges with this approach:

Naive Bayes assumes that all attributes are equally important [4]. In case of text mining, every word is an attribute and count of occurrences of that word is the value of our attribute. As we are dealing with free-flowing text in English, stop words cannot be considered to be 'important'. Thus, special care needs to be taken to make sure all stop words are removed. Thus, we added more stop words to the standard list of stop words. [1]

Steps to build the model:

We built the Naive Bayes model using scikit package available in Python [2][3]. This is our final model, hence input files to run the model are provided archive file Project2_code.zip

Input files: testMining.py, stopWords.txt

1. Take the input data set and divide it in training, testing and validation data sets with a ratio of 60/20/20 and store the data. [9][10]

2. Remove the stop words in every document, using the CountVectorizer function available in the scikit package. The stop words come from a text file called stop_words.txt [1][2]

3. For the training corpus, create the Term Frequency and the Inverse Term frequency matrix using the "TfidfTransformer()" and "fit_transform()" functions available in scikit package.[2]

4. Pass the training data classes and the TFIDF matrix of the training corpus to the Naive Bayes function "MultinomialNB()" in scikit package which creates the Naive Bayes classifier. [2]

5. This classifier is now run on the testing and validation data set to determine the accuracy of the model.

Instances of training data set:

We built 2 models, K-NN and Naive Bayes model. For both models, the cleaning process was the same. The following are 3 instances of training data set we used to classify, removing the attributes.

Note: Stop Words are not removed, they are not explicitly considered while creating the bag of words using CountVectorizer() in scikit package in python as part of building our model.[2] We have also included 5 instances in 'DataInstances' folder in archive Project2_code.zip[6]

alt.atheism/51122:

```
In article <1993Apr5.091139.823@batman.bmd.trw.com>
jbrown@batman.bmd.trw.com writes:

>> Didn't you say Lucifer was created with a perfect nature?
>
>Yes.
>

Define perfect then.


>> I think you
>> are playing the usual game here, make sweeping statements like omni-,
>> holy, or perfect, and don't note that they mean exactly what they say.
>> And that says that you must not use this terms when it leads to
>> contradictions.
>
>I'm not trying to play games here.  But I understand how it might seem
>that way especially when one is coming from a completely different point
>of view such as atheism.
>
```

alt.atheism/51123:

```
NNTP-Posting-Host: punisher.caltech.edu

 arromdee@jyusenkyou.cs.jhu.edu (Ken Arromdee) writes:

 >>The motto originated in the Star-Spangled Banner.  Tell me that this has
 >>something to do with atheists.
 >The motto _on_coins_ originated as a McCarthyite smear which equated atheism
 >with Communism and called both unamerican.

 No it didn't.  The motto has been on various coins since the Civil War.
 It was just required to be on *all* currency in the 50's.

 keith
```

alt.atheism/51120:

```
X-Newsreader: rusnews v1.01

dmn@kepler.unh.edu (...until kings become philosophers or philosophers become kings) writes:
>      Recently, RAs have been ordered (and none have resisted or cared about
> it apparently) to post a religious flyer entitled _The Soul Scroll: Thoughts
> on religion, spirituality, and matters of the soul_ on the inside of bathroom
> stall doors. (at my school, the University of New Hampshire) It is some sort
> of newsletter assembled by a Hall Director somewhere on campus. It poses a
> question about 'spirituality' each issue, and solicits responses to be
> included in the next 'issue.' It's all pretty vague. I assume it's put out
> by a Christian, but they're very careful not to mention Jesus or the bible.
> I've heard someone defend it, saying "Well it doesn't support any one religion.
> " So what??? This is a STATE university, and as a strong supporter of the
> separation of church and state, I was enraged.
>
>      What can I do about this?

It sounds to me like it's just SCREAMING OUT for parody.  Give a copy to your
friendly neighbourhood SubGenius preacher; with luck, he'll run it through the
mental mincer and hand you back an outrageously offensive and gut-bustingly
funny parody you can paste over the originals.

I can see it now:

                              The Stool Scroll
         Thoughts on Religion, Spirituality, and Matters of the Colon

                       (You can use this text to wipe)


mathew
```

Results and Evaluations:

We evaluated our model based on the following factors:

Confusion Matrix: As per our logic of using Term Frequency Inverse Document Frequency[5] to determine the classification, if result of classifier value for a document is incorrect, we expect the incorrect value to have a value closest to the actual class value(a false positive). For example, rec.sport.baseball being classified as rec.sport.hockey. However, we observed that the false positives are spread over the all the other topics.

From the confusion matrix for Naive Bayes, we observed that the class of false positive values are very close to their actual class value. For example, most of the false positive values for topic "rec.sport.hockey" belong to topic "rec.sport.baseball". Of the 200 documents for "rec.sport.hockey" , 182 are classified correctly as "rec.sport.hockey" whereas all 5 are classified as "rec.sport.baseball ", while the remaining classifications are spread out.

However, for the K-NN model, we saw that false positives are spread out over the entire confusion matrix, with no set pattern. For example, for " sci.electronics", the 12 the row of the confusion matrix, we observe that the false positive values are spread over all topics.

Time Complexity: K-NN approach turned out to be very slow, taking about 20 min to build the model and the train and validate the data. Naive Bayes algorithm worked much faster than K-NN algorithm. While K-NN model took about 20 minutes to train, test and validate the entire data set, Naive Bayes took only about 5 minutes.

Accuracy: In terms of accuracy, Naive Bayes gives us a much higher accuracy rate than K-NN. The accuracy in the Naive Bayes Model is about 84% whereas, K-NN has an accuracy of about 65% for the same data set.

Space: Both Naive Bayes and K-NN algorithm take the same amount of space, as both models need to store the TFIDF matrix for the training corpus.

Features for which our final model did not work:

Dual-topic classification: The logic of our Naive Bayes Algorithm model is to find a news group topic which is most probable to "owning" the incoming new data instance. Naive Bayes works on the conditional probability concept.[4] The new data instance text is evaluated and the probabilities of this text belong to one of the news group topics, given that other words of the data instance also match those of the other words in the same topic is found. The newsgroup topic which gives the highest probable match with the new data instance is chosen as the classifier value for that data instance.

Thus, because we are using word counts to find such probabilities in our model, there will be cases where the data instances are mapped to a topic which is closest to the real newsgroup topic.

Decision for Final Model: Model 2, Naive Bayes

There were two major factors we considered while deciding the model - **Accuracy and Time**

K-NN model has an accuracy of 65%. A good accuracy value depends on many factors, like the training data available, the kind of testing data, algorithm used, efficiency of the algorithm etc. However, for this project, the given data set is sufficient for training the model. Moreover, the testing and validation dataset is also known, and they are not very different from the training data set.

Given the above factors, we decided that 65% accuracy is not 'good enough' for the given data set. In addition to low accuracy, it also took about 20 min to train, test and validate the model, which is too high. Hence, we decided to go with our second data model - Naive Bayes. However, we wanted to make sure that the accuracy of Naive Bayes is not because of our model being 'over - trained'[10]. We evaluated the confusion matrix for the same.

Based on evaluation of the confusion matrices for both the models, we see that for Naive Bayes, the false positives occur due 'dual topics' present. There is no specific pattern of "bias" towards any specific topic. Hence, we concluded that the high accuracy rate for Naive Bayes in not a result of model being over-trained. For K-NN algorithm, the confusion matrix shows that documents belonging to same topic, if incorrectly classified, are found to be classified in varied range of other topics.

Moreover, Naive Bayes model works much faster than K-NN model. Hence, we decided to use Naive Bayes as our final classifier. Thus, we decided to use Naive Bayes as our final model.

Conclusions:

We built 2 models for this project, first model using K-Nearest Neighbor (K-NN) Algorithm and second using the Naive Bayes Algorithm.

For the first model, we obtained a low accuracy rate of 65%. Moreover, in K-NN, every time a new data instance occurs, the algorithm needs to find distance of the new data instance from all its K nearest neighbors. After this, the 'voting' process occurs and the majority class is decided as the class value. Thus, time complexity for the same is very high. [11][8]

For the second model, we obtained a higher accuracy rate of 84%. and the model is built much faster than K-NN. We learnt that the reason Naive Bayes is a fast learning algorithm is because once it calculates the required probabilities, all it has to do is behave like a 'look-up' for any incoming new data instance and find the closest matches. 5][8]

Looking back, we think if we would have spent more time analyzing the data, we could have created some way to deal with words which may be able to classify the document immediately. For example, one way we could have been to create an "important words" list, just like the stop words list. Doing that, we would have given more weight to those words which are most relevant to the topic. For example, if our important words list has "angel", "soul", "God" and if we find some way to give more importance to these words while building our model during our training phase, do the same for any new data instance and then compare, our classification accuracy might have improved further.

One step we think was unnecessary was building our own K-NN algorithm. Looking back, we probably could have directly used the tools available to perform these functions. However, we did gain a few insights while implementing K-NN on our own, like the "important words" concept.

Overall, we were successfully able to build a good text classifier model with good accuracy and fast speed. As our model creates the Term Frequency- Inverse Document Frequency matrix [6]solely based on the words occurring in the texts of training document set, our solution is generalizable. Our solution should work for other kinds of data, provided data fed to the model in the training phase is free-flowing text data.

References:

1. For Stop words: RIT mycourses/CSCI.720.03 - Big Data Analytics /Resources /StopWords

2. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

3. Scikit-learn: API design for machine learning software: experiences from the scikit-learn project, Buitinck et al., 2013.

4. Slides on Naive Bayes Algorithm, by Dr. Trudy Howles, from RIT mycourses/CSCI.720.03 - Big Data Analytics /Materials/NaiveBayes-2016.pdf

5. Slides on Text Mining Algorithm, by Dr. Trudy Howles, from RIT mycourses/CSCI.720.03 - Big Data Analytics /Materials/TextMining-2016.pdf

6. 20 newsgroups Dataset credit : Carnegie Mellon

7. Project Report Format credit : Dr Trudy Howles, from RIT mycourses/CSCI.720.03 - Big Data Analytics /Projects/Project2-Text

8. Author: Sebastian Raschka, Article Title: Naive Bayes and Text Classification – Introduction and Theory, Written on October 4, 2014. Accessed on November 14, 2016.

9. Stack Overflow: open source, author(s): Stack Overflow Community

http://stackoverflow.com/questions/8625991/use-python-os-walk-to-identify-a-list-of-files

10. Slides on Text Mining Algorithm, by Dr. Trudy Howles, from RIT mycourses/CSCI.720.03 - Big Data Analytics /Materials/TrainTestValid-2016.pdf

11. Slides on Text Mining Algorithm, by Dr. Trudy Howles, from RIT mycourses/CSCI.720.03 - Big Data Analytics /Materials/Classification-NearestNeighbor.pdf